Approximating Material Interfaces during Data Simplification

David E. Sigeti* Benjamin F. Gregorski†
John Ambrosiano * Gerald Graham *
Mark A. Duchaineau‡ Bernd Hamann † Kenneth I. Joy †

* Los Alamos National Laboratory
† University of California, Davis
‡ Lawrence Livermore National Laboratory

1 Introduction

We present a new method for simplifying large data sets that contain material interfaces. Material interfaces embedded in the meshes of computational data sets are often a source of error for simplification algorithms because they represent discontinuities in the scalar or vector fields over a cell. By representing material interfaces explicitly in a data simplification process, we are able to provide separate field representations for each material over a single cell and, thus, to represent the fields much more accurately. Our algorithm uses a multiresolution tetrahedral mesh supporting fast coarsening and refinement capabilities and error bounds for feature preservation. We represent a material interface or other surface of discontinuity as the zero set of a signed distance function. This representation makes it possible to maintain continuity of the surface across cell boundaries. It also makes it possible to represent more complex interface structures within a cell, such as T-intersections. Within a cell, a field is represented on either side of the surface of discontinuity by separate linear functions. These functions are determined by true and "ghost" values of the field at the vertices of the cell. By requiring that each vertex have only one ghost value for a given field and material, we are able to avoid introducing spurious discontinuities in the fields at cell boundaries. The use of linear functions determined by ghost values makes it unnecessary to divide the original cells in the mesh along the surface of discontinuity, avoiding the resultant introduction of complex cell types and field representations. It also decouples the field representation from the representation of the surface of discontinuity, making it easier to represent fields when the material interfaces are more complex. Both the signed distance function that defines the surface of discontinuity and the ghost values that determine the field representations are handled very simply during refinement and coarsening of the mesh ensuring that all spurious discontinuities can be avoided with a minimum of computation and programming effort. We have applied our algorithm to simplification of a test problem from a well known fluid dynamics code with excellent results. Graphical and numerical results are presented. Furthermore, our multiresolution representation can be applied to other kinds of surfaces, e.g. isosurfaces.

2 Multiresolution Tetrahedral Mesh

As the geometric basis for our simplification algorithm we use the subdivision of a tetrahedral mesh presented by Zhou et al [1]. This framework has an important advantage over other multiresolution spatial data structures such as an octree—it makes it easy to avoid introducing spurious discontinuities into our representations of fields. The way we perform the binary subdivision ensures that the tetrahedral mesh will always be a conformant, i.e, all edges in the mesh end at the endpoints of other edges and not in the interior of edges. The simplest representation for a field within a tetrahedral cell is just the unique linear function that interpolates field values specified at the cell's vertices. In the case of a conformant mesh, this natural field representation will be continuous across cell boundaries, resulting in a globally C^0 representation.

We have generalized the implementation presented by Zhou et al by removing the restriction that the input data needs to be given on a regular rectilinear mesh consisting of $(2^N+1)\times(2^N+1)\times(2^N+1)$ points. A variety of input meshes can be supported by interpolating field values to the vertices of the multiresolution tetrahedral mesh. In general, any reasonable interpolation procedure may be used. In some cases, the procedure may be deduced from the physics models underlying the simulation that produced the data set. In other cases, a general-purpose interpolation algorithm will be appropriate.

We construct our data structure as a binary tree in a top-down fashion. Data from the input data set, including grid points and interface polygons, are assigned to child cells at the time that their parent is split.

The other basis for our algorithms is the ROAM system, described in [2]. ROAM uses priority queue-driven split and merge operations to provide optimal real-time display of triangle meshes for terrain rendering applications. The tetrahedral mesh structure used in our framework can be regarded as an extension to tetrahedral meshes of the original ROAM data structure for triangle meshes.

Since our data structure is defined recursively as a binary tree, a representation of the original data can be computed in a preprocessing step, and we can utilize the methods developed in ROAM to efficiently select a representation that satisfies an error bound or a desired cell count. This makes the framework ideal for interactive display.

Strict L^∞ error bounds are incorporated into the subdivision process, see Section 5 below.

3 Representing Material Interfaces

In the class of input datasets with which we are working, material interfaces are represented as triangle meshes. In the case that these triangle meshes are not known, they are extracted from volume fraction data by a material interface reconstruction technique described in [3] and [4] (The

*{ambro,ggraham,sigeti}@lanl.gov
†{gregorsk,hamann,joy}@cs.ucdavis.edu
‡{duchaineau1}@llnl.gov

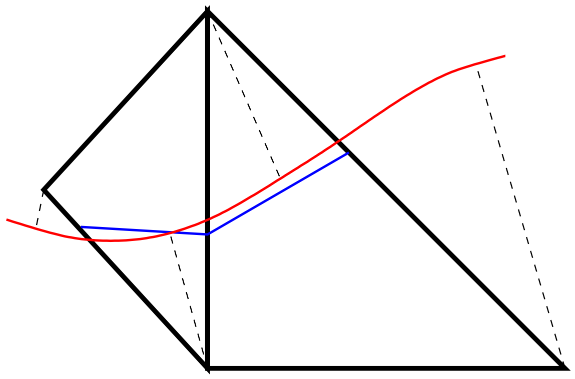

Figure 1: True and approximated interfaces.

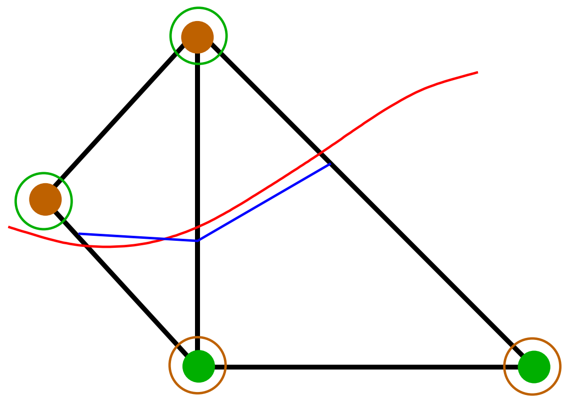

Figure 2: Ghost values.

volume fractions resulting from numerical simulations indicate what percentages of which materials are present in each cell.). Such an interface reconstruction technique produces a set of crack-free triangle meshes and normal vector information that can be used to determine on which side and in which material a point in space lies.

Within one of our tetrahedra, an approximate material interface is represented as the zero set of a signed distance function. Each vertex of a tetrahedron is assigned a signed distance value for each of the material interfaces in the tetrahedron. This value is simply the minimum distance from the vertex to the interface. The sign of the distance is given by the side of the interface on which the vertex lies.

Figure 1 shows a two-dimensional example of two triangles forming a conformant mesh, crossed by an interface (shown in red). The minimum distances from the vertices of the triangles to the interface are shown as dotted lines. The distances for vertices on one side of the interface (say, above the interface) are assigned positive values and those on the other side are assigned negative values. These signed distance values at the vertices will then determine linear functions in each of the triangles and the approximated interface (shown in blue) will be the zero set of these linear functions. Because the mesh is conformant, the linear functions in the two triangles will agree on their common side, and the zero set will be continuous across the boundary. The situation in three dimensions is analogous, with the word "triangle" replaced by "tetrahedron".

We note that, in order for the interface representation to be continuous across cell boundaries, it is necessary both that the mesh be conformant and that each vertex have at most one signed distance value for each interface.

The signed distance values for a vertex are computed when the vertex is created in a split operation. When searching for the point on the true interface that is closest to the vertex, it is possible to restrict attention those cells that share the edge being split. This makes the computation very efficient for the great majority of vertices.

4 Representing Discontinuous Fields

Once we have approximated the interfaces within a cell, we must decide how to represent fields on either side of the interface. Our algorithm represents the discontinuity by constructing a linear field representation for each material in the cell. In order to specify these representations, each of the vertices in a cell must have a distinct field value for each material in the cell. When a vertex does not lie in a given material, the field value associated with that material is called a *ghost value*.

The use of ghost values is illustrated in Figure 2. The material on the upper side of the interface is represented by brown and the material on the lower side is represented by green. The two upper vertices lie in the brown material and, thus, have regular values for the field in the brown material. These values are indicated by the the solid brown circles. The empty green circles indicate that these vertices require ghost values for the green material. Similarly, the lower circles lie in the green material and, thus, have regular values for the field in the green material (solid green circles) and require ghost values for the field in the brown material (empty brown circles). Once we have such ghost values, we can define linear representations for the field in the two regions by the usual interpolation. If we maintain a conformant mesh and assign only a single ghost value for a given material to a vertex, then our field representation will be discontinuous where it should be (across the interface) but not across cell boundaries (which would be a spurious discontinuity). Once again, the situation in three dimensions is analogous, with the word "triangle" replaced by "tetrahedron".

In our current implementation, we choose as the ghost value for a given vertex, field, and material the value of the field at the point in the material that is closest to the cell. These points are, of course, exactly the points that were used to determine the distance map that defines the approximation to the interface.

The ghost values for a vertex are computed when the vertex is created during the tetrahedral refinement process.

5 Error Bounds and Refinement Strategy

The error bounds employed in our framework are similar to the nested error bounds used in the ROAM system. Each cell has two associated kinds of error values, field errors and material interface errors.

Field errors are first calculated for leaf cells and are then propagated up the hierarchy. So far, we have only worked with input data sets that may be considered to consist of discrete grid points. In this case, the computation of error bounds for leaf cells is straightforward—the error for a leaf cell is simply the maximum of the errors associated with all the grid points from the input data set that it contains. When fields in the input data set are considered to have values over finite volumes, the computation of leaf cell errors will be more complex.

The field error e_T for a non-leaf cell is computed from the errors associated with its two children according to:

$$e_T = \max\{e_{T_0}, e_{T_1}\} + |z(v_c) - z_T(v_c)| \qquad (1)$$

where e_{T_0} and e_{T_1} are the errors of the children; v_c is the vertex that splits the parent into its children; $z(v_c)$ is the field value assigned to v_c; and $z_T(v_c)$ is the field value that the parent assigns to the spatial location of v_c, equivalently, $z_T(v_c) = \frac{1}{2}(z(v_0) + z(v_1))$, where v_0 and v_1 are the vertices of the parent's split edge. This error bound is *nested* in the sense that the error of a child is guaranteed not to be greater than the error of the parent.

The material interface error associated with a leaf node is the maximum of the errors associated with each of the interfaces in the node. For each interface, the error is the maximum distance between the approximate representation of the interface in the cell and those polygons that define the true interface and which are contained in the cell.

We initially refine our mesh to meet a user-determined error bound on the location of interfaces. The mesh is then further refined, using the ROAM algorithms, to minimize the error in a given field consistent with a given tetrahedron count.

6 Results

We have tested our algorithm on a data set resulting from a simulation of a hypersonic impact between a dense projectile and a less dense metal block. The simulation uses a logically rectilinear mesh of dimensions 32x32x52. For each cell, the density and pressure values are available, as well as the per-material densities and volume fractions. The physical dimensions in x, y, and z directions are [0,12] [0,12] and [-16,4.8].

There are three materials in the simulation: the projectile, the block, and empty space. The interface between the projectile and the block consists of 38 polygons, the interface between the projectile and empty space consists of 118 polygons and the interface between empty space and the block consists of 17574 polygons.

Figures 3 shows a cross section view of the mesh created by a cutting plane. The black lines are the original interface polygons intersected by the plane, and the magenta lines are our approximation to the interface. The interface approximation error is 0.15. An error of 0.15 means that all of the vertices in the original material interface meshes are no more that a physical distance of 0.15 from their associated interface approximation. This is equivalent to an error of (0.5 - 1.5)% when considered against the physical dimensions. A total of 3174 tetrahedra were required to approximate the interface to an error of 0.15. The overall mesh contained a total of 5390 tetrahedra. A total of 11990 tetrahedra were required to approximate the interface to an error of 0.15 and the density field to an error of 3. The maximum field approximation error in the cells containing material interfaces was 2.84 and the average field error for these cells was 0.007. These error measurements indicate that separate field representations for the materials on either side of a discontinuity can accurately reconstruct the field.

Figures 4 and 5 compare density fields generated using linear interpolation of the density values to fields generated using separate field representations on either side of the discontinuity. Figure 5 shows that using explicit field representations in the presence of discontinuities can improve the quality of the field approximation. This can be seen in the flat horizontal and vertical sections of the block where the cells approximate a region that contains the block and empty space. In these cells, the use of explicit representations of the discontinuities leads to a very accurate representation of the density field. The corresponding field representations using linear interpolation, shown in Figure 4, do a very poor job of capturing the discontinuities. Furthermore, Figure 5 captures more of the dynamics in the area where the projectile is entering the block (upper left corner). The linear interpolation of the density values in the region where the projectile is impacting the block smoothes out the density field, and does not capture the distinct interface between the block and the projectile. Figure 6 shows the density field from Figure 5 with our approximation to the interface and without the cell outlines.

7 Conclusions and Future Work

We have presented a simplification method for scientific data sets that explicitly represents material interfaces in mesh cells. Our algorithm constructs an approximation that can be used in place of the original data set for visualization purposes. Explicitly representing the material and implicit field discontinuities allows us to use multiple field representations to better approximate the field within each cell. The use of the tetrahedral subdivision allows us to generalize our algorithm to a wide variety of data sets and to support interactive level-of-detail exploration and view-dependent simplification. Future work will extend our error calculations to support complex input cell types such as tetrahedra and curvilinear hexahedra. Our current ghost value computation assumes that the field is constant on the other side of the interface. Higher-order extrapolation methods should be investigated for ghost value computation to determine if a superior field approximation can be obtained. Similarly, material interfaces are defined by approximations based on linear functions. The tradeoff between cell count and higher-order approximation methods should be investigated to determine if a better approximation can be obtained without a great increase in computational complexity. Finally, we plan to apply our algorithm to more complex unstructured data sets.

References

[1] Y. Zhou, B. Chen, and A. Kaufman, "Multiresolution tetrahedral framework for visualizing regular volume data," in *IEEE Visualization '97* (R. Yagel and H. Hagen, eds.), pp. 135–142, IEEE Computer Society Press, 1997.

[2] M. A. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein, "ROAMing terrain: Real-time optimally adapting meshes," in *IEEE Visualization '97* (R. Yagel and H. Hagen, eds.), pp. 81–88, IEEE Computer Society Press, 1997.

[3] K. S. Bonnell, "On material boundary surfaces," Master's thesis, University of California at Davis, June 2000.

[4] K. S. Bonnell, M. A. Duchaineau, D. R. Schikore, B. Hamann, and K. I. Joy, "Constructing material interfaces from data sets with volume-fraction information," in *IEEE Visualization 2000*, 2000. to appear.

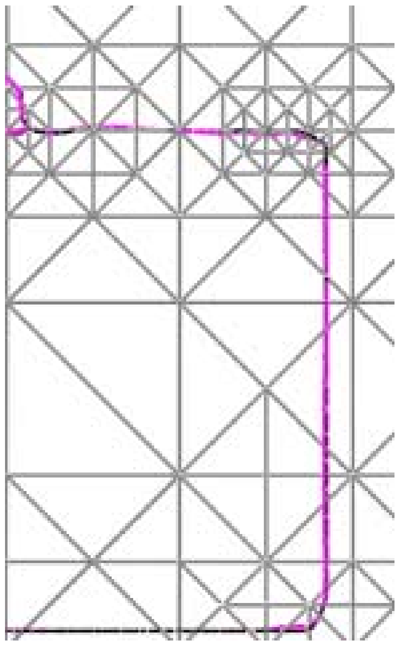

Figure 3: Cross section of the tetrahedral mesh showing the original interfaces and interface approximations.

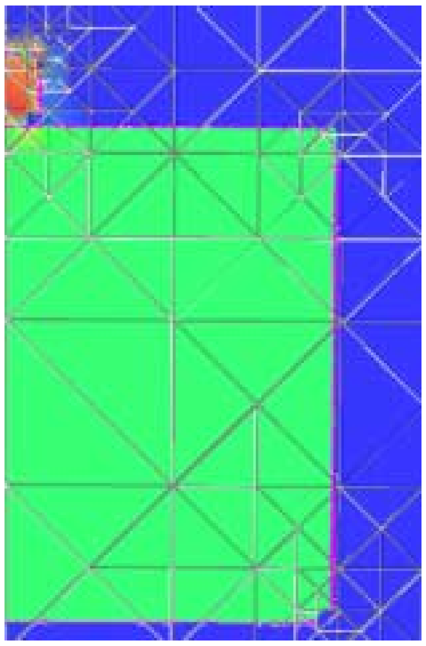

Figure 5: Density field using explicit interface representations and separate field representations (interface error = 0.15).

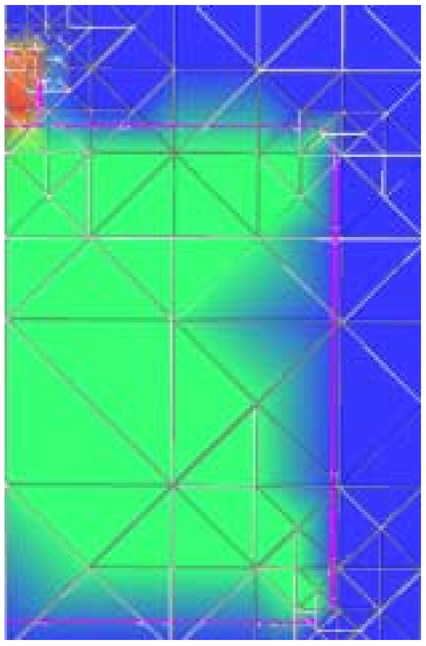

Figure 4: Density field using linearly interpolated density values (interface error = 0.15).

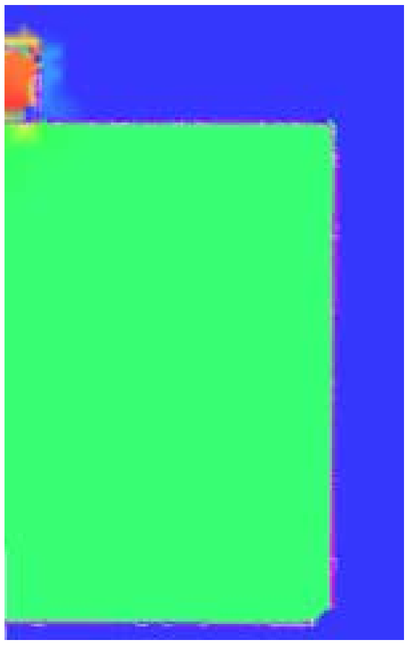

Figure 6: Figure 5 without the cell outlines.